Visual Programming Languages : More Than Just a Pretty Face

Marat Boshernitsan

Michael Downes

Contents

- Motivations
- History
- Theory
- Taxonomy
- Features and Implementation Issues
- Example languages
- Conclusions

Motivations

- Many people think in pictures.
- Textual programming languages have proven to be difficult for many people to learn to use effectively.
- Some applications are very well-suited to graphical development approaches. (e.g. scientific visualization and simulation creation)

History

- Sketchpad (I.B. Sutherland, 1963)
  - simple constraint-based graphics system
  - allowed user to display and manipulate abstractions, like constraints on geometry etc.
  - 1965 - W. Sutherland created a dataflow language on the same machine (the TX-2), it allowed for visual creation, debugging, and execution of dataflow diagrams
- Pygmalion (D.C. Smith, 1975)
  - considered the starting point for much of modern VPL research
  - attempted to provide a method of programming which corresponded to the thought processes extant in creative thought
  - presented an icon-based programming paradigm
  - the programmer instructed the system to "remember" his/her operations in order to create text programs written in a simple assembly-like subset of Smalltalk (i.e. - programming by example)

Theory

- based on work by Chang
- definitions
  - **iconic system**: structured set of related icons
  - **iconic sentence (visual sentence):** spatial arrangement of icons from iconic system
  - **visual language**: set of iconic sentences constructed with given syntax and semantics
  - **syntactic analysis of visual languages (spatial parsing):** analysis of an iconic sentence to determine the underlying structure
  - **semantic analysis of visual languages (spatial interpretation):** analysis of an iconic sentence to determine the underlying meaning
- kinds of icons
  - **process icons**: express computations (primitives)
  - **elementary object icons**: identify primitive objects (primitives)
  - **composite object icons**: identify composite object (constructed from elementary object icons using construction operators)

- formation of visual sentences:
  - conventional programming languages: textual tokens are concatenated to form a textual sentence (concatenation is implicit)
  - visual programming languages: icons correspond to tokens; construction rules are explicit (part of the language):
    - horizontal concatenation
    - vertical concatenation
    - spatial overlay
- syntactic analysis:
  - picture grammars, graph grammars, ...
  - result: good old parse tree
- semantic analysis:
  - similar to conventional programming languages (attribute grammars, etc.)
- recently some effort went into formalizing VLs
  - e.g. lambda-calculus (Citrin95), types & type inferences (Burnett93)

Taxonomy

- based on work by Chang, Shu, and Burnett
- purely visual systems in which icons or other graphical representations are manipulated to create a program which is then debugged and executed in the same visual environment
  - e.g. VIPR, Prograph, PICT/D, Cube
- hybrid text and visual languages in which programs are created visually and then translated into an underlying textual language or which involve the use of graphical elements in an otherwise textual language
  - e.g. Rehearsal World, C^2, work by Erwig
- programming by example systems
  - e.g. Rehearsal World, Pygmalion
- constraint-oriented and physical simulation systems
  - e.g. ThingLab, ARK
- form-based languages which use visualizations descended from a spreadsheet metaphor
  - e.g. Forms/3

- VPL Orphans
  - algorithm animation systems
    - e.g. BALSA
  - user-interface design systems
    - e.g. the Microsoft visual buzzword herd
- Commercial successes
  - Prograph
  - LabVIEW
  - IRIS Explorer

Language Issues

- control flow:
  - imperative approach: flow diagrams
  - declarative approach: object dependencies, pattern-matching
- level of procedural abstraction:
  - high level: not possible to maintain and write entire program in given language; e.g. software engineering tools, scientific visualization
  - low level: express fine-grained logic but no means for generalization; e.g. domain-specific visual languages
- data abstraction:
  - consistency with visual paradigm: no underlying textual representation
  - visual definition of abstract data types
  - objects resulting from such definitions are visual

VIPR (Visual Imperative PRogramming)

- **Who** : Citrin, Doherty, and Zorn (1994 - present)
- **What** : object-oriented completely visual language
- **Why** : motivations include producing an object-oriented language that's easy to learn and use
- control-flow:
  - pipe metaphor for computation, state travels down pipe and rings merge at each step in computation
- level of procedural abstraction:
  - low and high-level programming analogous to C++
- data abstraction:
  - it is a programming language, not a program representation, there are graphical rewrite rules so a VIPR program can be executed in the same visual environment in which it is created
- semantics similar to C++ but can be defined entirely by graphical transformation rules
- statically scoped and typed; includes inheritance, polymorphism, dynamic dispatch

VEX (Visual EXpressions)

- visual representation of lambda expressions
- expression-oriented component of VIPR
- motivation : to provide a more effective method for teaching lambda calculus than by textual methods
- paper presents pure untyped lambda calculus with discussion of how to extend it
- includes graphical definitions of α-conversion, β-reduction, and η-reduction

ARK (Alternate Reality Kit)

- **Who** : R.B. Smith (1986 - 1992)
- **What** : 2D animated environment for creating interactive simulations, implemented in Smalltalk-80
- **Why** : to teach users about the fundamental laws of physics and allow non-expert programmers to develop interactive simulations
- objects have visual representation, mass, and velocity
- laws of nature are themselves objects which can be manipulated and changed
- main components : interactors, message boxes, buttons, menus, objects, the hand
- multiple alternate realities are contained within a meta-reality (analogous to windows and desktop)
- tension between magic and literalism
- underlying Smalltalk objects can be accessed through Representatives
- recent work has involved developing a distributed version called SharedARK

Prograph

- **Who** : T. Pietryzkowski & P. T. Cox, now commercialized by Pictorius, Inc.
- **What** : object-oriented pictorial programming environment
- **Why** : pictoriality can be used to replace familiar nested control structures by multiplexes and Prolog-like pattern-matching
- control-flow:
  - combine declarative and imperative semantics: procedures described as control-flow diagrams; method invocation is based on pattern-matching
- level of procedural abstraction:
  - low-level programming done via method definitions
  - high-level programming accomplished by combining methods into classes which are in turn are combined into class libraries (hierarchies)
- data abstraction:
  - each abstract data type is encapsulated in a class
  - objects are instantiated from classes (similar to class-bases languages)

Forms/3

- **Who** : Margaret Burnett, et. al
- **What** : form-based object-oriented programming language
- **Why** : borrow the spreadsheet ideas of cell and formulae as the basic way that programming is done
- control flow:
  - combines declarative semantics with time dimension (vectors in time)
- level of procedural abstraction:
  - low-level programming done via formulae that are associated with each data cell
  - high-level programming is accomplished by collecting cells together in a form
- data abstraction
  - provides visual abstract data types (VADT) -- forms that consist of components, operations, graphical representations, and behaviors
  - correspond to prototype-based languages

Conclusions

- The field of visual programming languages is starting to mature as evidenced by the flow of current research toward formalizing the theoretical foundations of VPLs and developing standardized classifications for VPLs.
- It is not worthwhile to try to eschew text entirely. (e.g. Atomic operations, like addition, can be represented graphically in VIPR, but programs are easier to read and write when such operations are presented as text.)
- VPL research provides an interesting mix of problems in computer graphics, programming languages, and human-computer interaction.
- So far, the most successful visual systems have been those which target a well-defined application area. (e.g. Explorer, BALSA, ARK, LabVIEW)